Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Caustics Rendering and Synthetic Data

Bachelor Thesis

2024
Michal Kubirita

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Caustics Rendering and Synthetic Data

Bachelor Thesis

Study Programme:	Applied Computer Science
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	Mgr. Lukáš Gajdošech
Consultant:	doc. RNDr. Martin Madaras, PhD.

Bratislava, 2024
Michal Kubirita

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Michal Kubirita
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Synthetic Dataset Rendering with Parametric Transparent and Translucent Objects
Generovanie syntetických dát s parametrickými priesvitnými a priehľadnými objektami

Anotácia: K tréningu robustnej neurónovej siete potrebujeme prístup k trénovacím dátam. Konvolučná neurónová sieť vie byť natrénovaná metódami hlkbokého učenia s použitím syntetických, alebo reálnych anotovaných dát, za účelom spracovania mračien bodov ako napr. Filtrovanie alebo odhad pózy objektu. Pokiaľ nie sú k dispozícii reálne a anotované dáta, syntetické datasety môžu byť renderované a použité na tréning. Hlavným cieľom práce bude navrhnutie renderovacieho systému na renderovanie scén zložených z priesvitných a priehľadných predmetov. Vygenerované dáta budú neskôr použité na tréning neurónovej siete na spracovanie mračien bodov. Priehľadné a priesvitné objekty by mali byť parametrizované za účelom randomizácie v generovanej scéne. V optimálnom prípade, renderovanie chceme mať v reálnom čase, preto chceme použiť Unreal Engine. Alternatívne sa môže použiť Blender s LuxCore rendererom.

Cieľ:
- Presúmať možnosti renderovacích a herných enginov ako Unreal Engine a LuxCoreRender v Blendery za účelom renderovania priehľadných a priesvitných objektov
- Prioritizujte rýchlejšie renderovanie, ideálne v reálnom čase, preto sa zamerajte primárne na Unreal Engine
- Vytvorte parametrický model transparentných a translucentných objektov, ktoré sa môžu parmetrizovať cez skript a vkladať do scény
- Renderujte scénu s kaustikami, s parametrickými modelmi, prípadne s existujúcimi priesvitnými a priehľadými modelmi z UE a porovnajte výsledky z UE s výsledkami z Blenderu

Literatúra:
Rui Wang, Wei Hua, Yuchi Huo, Hujun Bao 2022, Real-time Rendering and Editing of Scattering Effects for Translucent Objects, https://doi.org/10.48550/arXiv.2203.12339
Unreal Engine documentation, https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/HowTo/Transparency/
Unreal Engine forums, https://forums.unrealengine.com/t/transparent-materials-render-infront-of-closer-translucent-materials/278592
Unreal Engine RTX branch, https://developer.nvidia.com/game-engines/unreal-engine/rtx-branch

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

https://forums.unrealengine.com/t/inside-unreal-dlss-and-rtxgi-with-nvidia/149916

Kľúčové slová: synthetické dáta, priehľadné a priesvitné objekty, virtuálna scéna, Unreal Engine

Vedúci: Mgr. Lukáš Gajdošech
Konzultant: doc. RNDr. Martin Madaras, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 04.10.2023

Dátum schválenia: 04.10.2023 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Michal Kubirita
Study programme: Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study: Computer Science
Type of Thesis: Bachelor´s thesis
Language of Thesis: English
Secondary language: Slovak

Title: Synthetic Dataset Rendering with Parametric Transparent and Translucent Objects

Annotation: Training of robust neural networks is based on access to the training data. A convolutional neural network can be trained in a deep learning manner using synthetic or real captured annotated data to perform point cloud processing tasks, such as filtering the point clouds or pose estimation of an object. If there is no available real annotated training dataset a synthetic one can be rendered and used. The main scope of the thesis would be to propose a rendering pipeline for synthetic scans composed of transparent and translucent objects. The generated data will be later used to train a neural network for point cloud processing. The translucent and transparent objects should be parametrized in order to be randomized in the scene. In an optimal scenario, the rendering of the scene should be performed in real-time, therefore Unreal Engine should be used. Alternatively, a Blender with LuxCoreRenderer can be used.

Aim: - Explore the possibilities of rendering and game engines such as Unreal Engine or Blender's LuxCoreRender for rendering translucent and transparent objects
- Prioritize faster, real-time rendering, therefore focus mainly on Unreal Engine
- Create a parametric model of translucent and transparent objects that can be parametrized and added to a scene using a script
- Render the scene with all the caustics, with parametric models and assets from UE and compare the results rendered in UE with results from Blender

Literature: Rui Wang, Wei Hua, Yuchi Huo, Hujun Bao 2022, Real-time Rendering and Editing of Scattering Effects for Translucent Objects, https://doi.org/10.48550/arXiv.2203.12339
Unreal Engine documentation, https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/HowTo/Transparency/
Unreal Engine forums, https://forums.unrealengine.com/t/transparent-materials-render-infront-of-closer-translucent-materials/278592
Unreal Engine RTX branch, https://developer.nvidia.com/game-engines/unreal-engine/rtx-branch
https://forums.unrealengine.com/t/inside-unreal-dlss-and-rtxgi-with-nvidia/149916

Keywords: synthetic dataset, translucent and transparent objects, virtual scene, Unreal Engine

Supervisor: Mgr. Lukáš Gajdošech

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Consultant:	doc. RNDr. Martin Madaras, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	04.10.2023
Approved:	04.10.2023

doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: I am sincerely grateful for the unwavering moral support from my friends.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Contents

1 Background **1**

1.1 Rendering Techniques 1

1.2 Caustics 5

1.3 Unreal Engine 5

1.4 Blender 6

1.5 Related Work 7

2 Implementation **9**

3 Results **11**

Conclusion **13**

List of Figures

1.1 Ray casting diagram 2
1.2 Ray tracing diagram 4
1.3 Example of caustics beneath drinking glasses. 6

List of Tables

Chapter 1

Background

In computer vision transparent objects are a problematic subject. They may have indeterminate boundaries and end up imperfect on 3D scans due to reflection and refraction of light.

To render transparent objects realistically we need to consider physical interactions of light. Transparent materials behave differently compared to the opaque ones. Rays of light coming through them create illuminated effects called caustics.

This happens also inside transparent fluid bodies (e.g. water) but we will only focus on solid objects like bottles and glasses, which is referred to as mesh caustics [Yang and Ouyang, 2021].

The chapter covers related issues, known solutions and available tools.

1.1 Rendering Techniques

In 3D, rendering is the process of transforming a virtual 3D scene into a digital image. Rendering engines (renderers) use various methods to visualize a 3D scene and calculate lighting. We outline the basic terms.

Offline vs Online Rendering

Offline rendering is sometimes called pre-rendering, since its result is an already finished image, a render, that may have taken hours to create. Its main priority is photorealism with accurate physical lighting. It is used in films and static images demanding high image quality.

Online rendering is also called real-time or interactive rendering because of its high performance. Its main goal is performance - online renders take only a few milliseconds to make giving programs the ability to be visually interactive. It is used mainly in games.

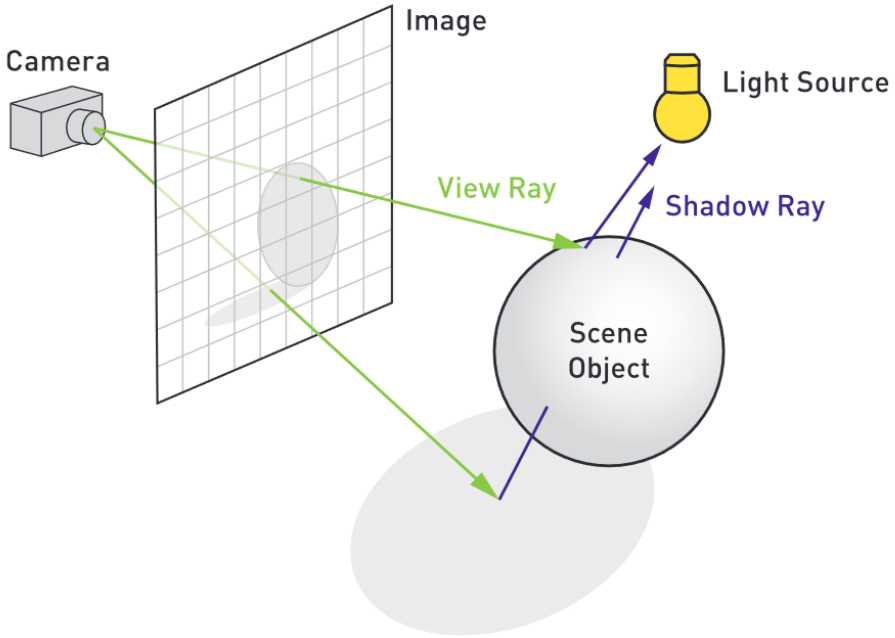

Figure 1.1: *Ray casting.* The diagram depicts hit point intersections of view rays. The hit point is shadowed if a shadow ray intersects an object on its way to a light source. *From Ray Tracing Gems by [Haines and Akenine-Möller, 2019].*

Rasterization

TODO

A common implementation of online rendering is rasterization. It is a process of rasterizing scene objects into pixels. The color of each pixel in the image is determined by pixel shaders.

Rasterization is able to generate many frames per second. However, it is very limited in producing accurate lighting.

Ray Casting

According to [Haines and Akenine-Möller, 2019], "ray casting is the process of finding the closest, or sometimes just any, object along a ray." In computer graphics, ray casting has many applications.

In the context of rendering, a ray is cast from the camera (also called the eye) through a pixel in the image plane. This type of ray is called a viewing ray and other times just a view ray [Glassner, 1989, Shirley et al., 2009, Haines and Akenine-Möller, 2019].

It travels until it hits the first object occluding the ray's path. For the purpose of shading the hit point, another ray can be cast into a light source to find out whether the object is in a shadow. For a visualization see figure 1.1.

Ray Tracing

Ray tracing is an implementation of offline rendering. Its history is well-documented in the book Ray Tracing Gems by [Haines and Akenine-Möller, 2019].

The algorithm uses ray casting recursively to determine the color of each pixel in the image plane. The recursion limit is set as a maximum number of light bounces. It is the number of hit points (intersections) one ray can possibly achieve.

A light source shines rays (photons) in many directions. Only a few of them can be caught by our eye and even then some may have taken an overly complex route. Thus, it is reasonably more efficient to shoot rays from the eye into the scene. These are guaranteed to hit the objects and other visible parts of the scene in our field of view (FOV).

The recursion occurs in the intersections. After hitting a surface, child rays may be cast to simulate reflection and refraction. The directions of child rays are determined by the surface's material. Each of these rays can also be tested for shadow occurrence. When a ray reaches its limit, the color of the pixel is estimated by accounting the viewing ray, shadow rays and reflection and refraction rays [Haines and Akenine-Möller, 2019]. For a visualization see figure 1.2.

A more advanced implementation casts many rays per pixel to make the object edges smoother, create soft shadows or effects like camera depth of field [Shirley et al., 2009]. This improvement has been proposed by [Cook et al., 1984] and is called distributed ray tracing.

Spawning just one reflection ray from the hit point on a surface creates a mirrored reflection. To achieve a glossy look of shiny and polished materials, we must spawn multiple rays in a conical shape around the hit point and blend their results [Haines and Akenine-Möller, 2019].

We need to realize that this process of ray tracing is reversed from the actual physical behaviour of lighting. For a comprehensive explanation of the fundamentals see the very first book about ray tracing [Glassner, 1989] and for more advanced and current topics see [Haines and Akenine-Möller, 2019, Marrs et al., 2021].

Forward and Backward Raytracing

TODO

Path Tracing

Path tracing is built upon the ray tracing algorithm. It uses Cook's sampling technique, each pixel is traced many times to light it more accurately.

Furthermore, matte surface lighting is computed differently, matching real-world behaviour. To compute reflection of a matte or a diffuse surface, reflections rays scatter

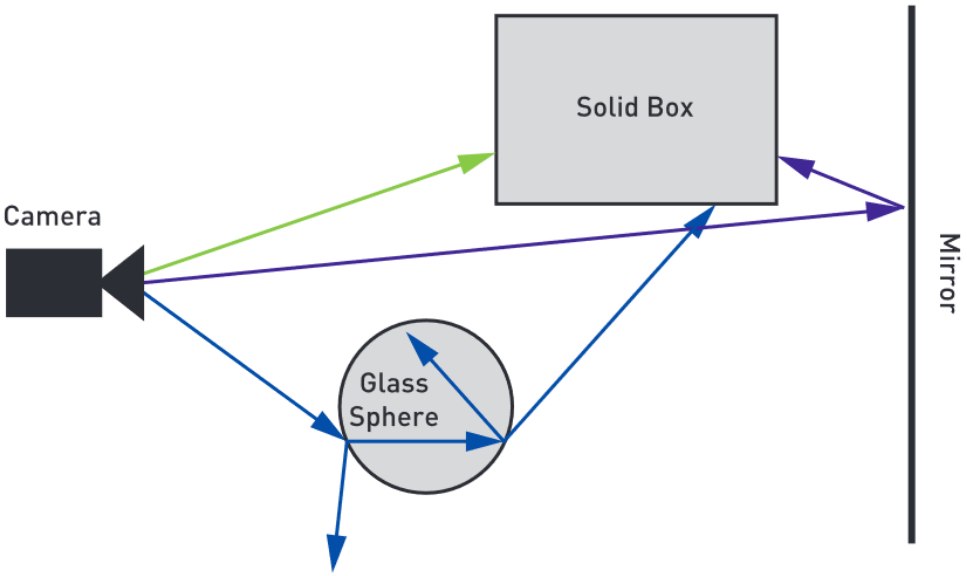

Figure 1.2: *Ray tracing.* The diagram depicts how a view ray forms child rays. The glass sphere as a translucent object reflects and refracts the rays, while the mirror only reflects them. As a result, the solid box affects reflections on the sphere and the mirror alike. *From Ray Tracing Gems by [Haines and Akenine-Möller, 2019].*

in all directions [Haines and Akenine-Möller, 2019].

When a ray hits this type of surface, a direction of a reflected child ray is chosen in a precise way MONTE CARLO. The child ray may reflect again, spawning a new ray recursively, until it finally reaches a light source. The more we repeat this process the more accurate the radiance of a pixel will be.

The sequence of ray intersections from the eye to a light source is called a path, thus the name path tracing. This style of rendering is credited to [Kajiya, 1986]. In the book Ray Tracing Gems [Haines and Akenine-Möller, 2019] say, "path tracing can, with proper care, give an unbiased result, one matching physical reality."

The Rendering Equation

The Cook's and Kajiya's rendering methods require a way to efficiently choose rays for sampling. They use Monte Carlo Integration algorithm with probability density functions, stochastically distributing rays of light to solve this issue [Haines and Akenine-Möller, 2019]. This technique has been called distributed ray tracing [Glassner, 1989].

Below is a transport equation in an energy-balanced form, in computer graphics commonly referred to as the rendering equation [Haines and Akenine-Möller, 2019].

$$L_0(P, \omega_0) = \int_{S^2} f(P, \omega_0, \omega_i)\ L_i(P, \omega_i)\ |\cos\theta_i|\ d\omega_i \tag{1.1}$$

L_0 is the light leaving from surface point P in direction ω_0. Function f represents a

bidirectional reflectance distribution function. L_i is the incoming light in direction ω_i calculated recursively. θ_i is angle between the surface normal and the incoming light direction. [Haines and Akenine-Möller, 2019]

By integrating through all the light sources and objects, we get the resulting ray's radiance.

Now we can clearly see why is the ray's color only approximated. The integral could be hardly solved for each pixel so choosing the best ray paths is necessary.

Bidirectional Path Tracing

TODO

Photon Mapping

TODO

Photon Splatting

TODO

1.2 Caustics

TODO

Caustic is a curve of light caused by reflection and refraction of light shined on a transparent or a translucent object [Yang et al., 2021]. Figure 1.3 shows an example of caustic patterns usually seen under drinking glasses.

Physically-accurate simulation of caustics is computationally expensive and simple rasterization rendering engines are not able to produce them. To simulate these effects we need to trace light bounces and thus use a path tracing or a raytracing engine.

Caustics can be rendered in a simplistic way too with methods of faking the light effects. However, to bridge the Sim2Real gap we aim to use hardware raytracing to achieve more accurate illumination for real-time renders.

1.3 Unreal Engine

TODO

Unreal Engine is a popular game engine developed by Epic Games. It receives regular updates oftentimes introducing state-of-the-art implementations of new features leading the game industry's technological progress.

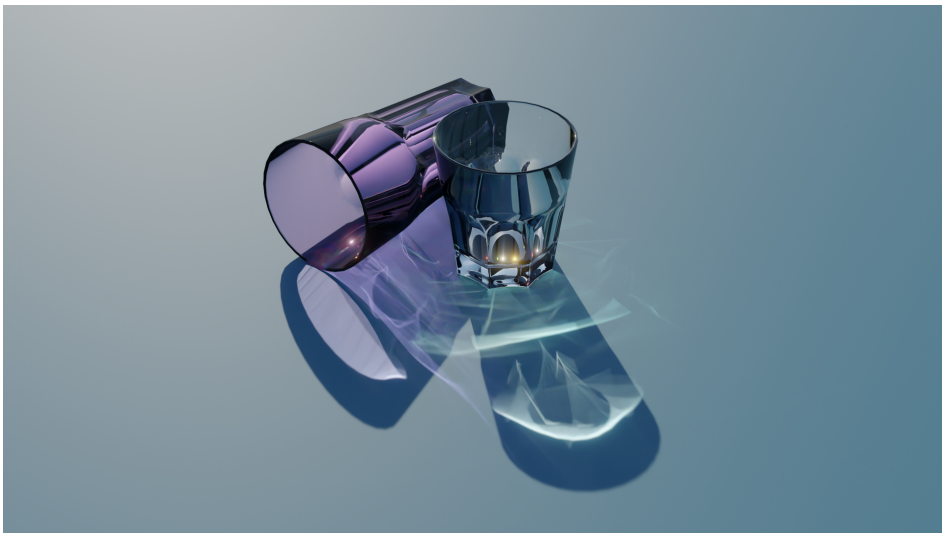

Figure 1.3: *Example of caustics beneath drinking glasses.* The reflected and refracted light rays concentrate into patches of light with intense bright edges sometimes forming cusp singularities. *Rendered in LuxCoreRender.*

It has a very large community consisting not only of game developers. With many years of its existence, the engine has evolved to an enormous platform for digital content creation which interests experts from various fields. Apart from making games it is used for architectural visualizations, automobile design, product design, simulations and films.

Nvidia's NvRTX Branches

Nvidia tightly cooperates with Epic Games to enhance rendering capabilities and introduce new features. Nvidia also implements its technologies into the engine such as DLSS, Frame Generation, Reflex and NRD.

Most of the experimental Nvidia features are accessible inside their forked Unreal Engine branch on GitHub. There is a specific branch for implementing caustics rendering which we aim to explore. [1]

1.4 Blender

TODO

Blender is a free open-source 3D program featuring tools for the whole 3D content-creation pipeline. It has an API for python scripting, making it easily extendable. It is cross-platform - runs on Linux, Windows and Macintosh.

[1] The Unreal Engine version we used is accesssible here.
`https://github.com/NvRTX/UnrealEngine/tree/NvRTX_Caustics-5.2`

1.5 Related Work

TODO

There are multiple existing tools regarding real-time rendering of synthetic data namely inside Unreal Engine. Its powerful rendering capabilities and ability to extend features through plugins shows why it is a popular choice among computer vision researchers.

SuperCaustics

This tool extends the Nvidia Unreal Engine Caustics 4.27 branch. Its purpose is to generate synthetic datasets of translucent objects with caustics. It is fully automated, generating random scenes in the editor and rendering them. [Mousavi and Estrada, 2021]

The tool is not functional, since the particular version of the branch cannot be installed properly. It relies on downloading dependencies from a server with forbidden access.

UnrealGT

UnrealGT is an Unreal Engine 4 plugin which provides the user with ability to generate generic image datasets inside the editor. [Pollok et al., 2019]

It works only in Unreal Engine 4 and is not compatible with Unreal Engine 5 and later versions because some game object classes have been refactored. That makes it impossible to compile this plugin inside Unreal Engine 5.

UnrealCV

UnrealCV is also an Unreal Engine plugin which helps computer vision researchers create synthetic data similarly as the previously mentioned works. Compared to the ones above, it runs on versions up to Unreal Engine 5.2.

It operates through a set of console commands inside the editor, allowing interactions with the virtual world. Alternatively, one can use their client API and connect to the editor from an external program [Weichao Qiu, 2017].

EasySynth

EasySynth is an Unreal Engine plugin for generating synthetic datasets for machine learning. It can export RGB, depth, normal and optical flow images. There is a tool for object segmentation which allows to also generate semantic images.

SynBin

A previous work explored possibilites of procedurally generating meshes inside Blender.

They proposed a pipeline for creating cardboard boxes with Geometry Nodes system. These models served as synthetic data for training neural network to 6D pose estimation task, which proved to be superior to previous attempts. [Kravár et al., 2023]

Chapter 2

Implementation

Chapter 3

Results

Conclusion

Bibliography

[Cook et al., 1984] Cook, R. L., Porter, T., and Carpenter, L. (1984). Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145.

[Glassner, 1989] Glassner, A. S. (1989). *An introduction to ray tracing*. Morgan Kaufmann.

[Haines and Akenine-Möller, 2019] Haines, E. and Akenine-Möller, T., editors (2019). *Ray Tracing Gems*. Apress. `http://raytracinggems.com`.

[Kajiya, 1986] Kajiya, J. T. (1986). The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150.

[Kravár et al., 2023] Kravár, P., Gajdoech, L., and Madaras, M. (2023). Novel synthetic data tool for data-driven cardboard box localization. In *International Conference on Artificial Neural Networks*, pages 565–569. Springer.

[Marrs et al., 2021] Marrs, A., Shirley, P., and Wald, I. (2021). *Ray tracing Gems II: next generation real-time rendering with DXR, Vulkan, and OptiX*. Springer Nature.

[Mousavi and Estrada, 2021] Mousavi, M. and Estrada, R. (2021). Supercaustics: Real-time, open-source simulation of transparent objects for deep learning applications. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 649–655. IEEE.

[Pollok et al., 2019] Pollok, T., Junglas, L., Ruf, B., and Schumann, A. (2019). Unrealgt: Using unreal engine to generate ground truth datasets. In *Advances in Visual Computing : 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I. Ed.: George Bebis*, page 670–682. Springer.

[Shirley et al., 2009] Shirley, P., Ashikhmin, M., and Marschner, S. (2009). *Fundamentals of computer graphics*. AK Peters/CRC Press.

[Weichao Qiu, 2017] Weichao Qiu, Fangwei Zhong, Y. Z. S. Q. Z. X. T. S. K. Y. W. A. Y. (2017). Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*.

[Yang et al., 2021] Yang, F., Wünsche, B. C., and MacDonald, B. (2021). Real-time caustics and dispersion on arbitrary surfaces in gpu-accelerated ray tracing. In *2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6.

[Yang and Ouyang, 2021] Yang, X. and Ouyang, Y. (2021). *Real-Time Ray Traced Caustics*, pages 469–497.